

Available online at www.sciencedirect.com**SciVerse ScienceDirect**

Procedia Computer Science 9 (2012) 364 – 372

Procedia
Computer Science

International Conference on Computational Science, ICCS 2012

A Fast GIS-tool to Compute the Maximum Solar Energy on Very Large Terrains

S. Tabik¹, A. Villegas, E. L. Zapata and L. F. Romero

Dept. Computer Architecture

University of Málaga

29080 Málaga, Spain

Abstract

This study presents a new functionality of Geographic Information Systems (GIS) to assess solar energy input on vast high resolution Digital Elevation Models (DEMs). This tool is able to find out 1) the maximum solar energy that can be captured on a surface situated at a determined height on each point of the DEM and 2) the optimal angles (i.e., slope and orientation) that allow capturing this maximum energy. Insulator: the open source high performance solar radiation model, we developed in a previous work, is used as baseline for this tool. Contrarily to most existent GIS tools, the proposed algorithm doesn't suffer memory limitations and is specially suitable for GPU-CPU heterogeneous systems. The experimental results show that the proposed algorithm is able to compute the maximum irradiation and optimal angles maps on large DEMs with high accuracy in very short times and showing a high scalability.

Keywords: GIS-tool, solar energy, DEMs, CPU-GPU heterogeneous architecture

1. Background and Motivation

Knowledge of solar energy input that can be collected at a given terrain is essential to maximize energy production output. There exists a large number of applications where the investment depends directly on the accuracy of this knowledge. Some useful applications in this context are:

1. To preliminary know the solar energy input and deduce the electric energy output that could be produced by a predefined photovoltaic panel if installed on a specific place [8, 9].
2. To find out the most suitable sites for the installation of a solar plant and determine the optimal slope and orientation of each one of their modules.
3. To preliminary know the quantity of solar energy that can be captured by a hypothetical solar plant. This is especially useful when a company or institution have to decide whether to invest in solar or eolic energy in a specific terrain and needs to compare the electric energy production of a hypothetical solar plant versus the energy output estimations of a hypothetical eolic plant in that specific place.

¹Correspondence author E-mail: stabik@uma.es

4. The need to determine the optimal slope and orientation angles of solar collectors dedicated to drying crops in emerging countries for different periods [10].

The last decade has known the development of new radiation models for calculating the incoming radiation on global areas represented by DEMs that fit in the main memory of single processor desktops. However, there exist few solar GIS-tools that respond to the applications enumerated previously due to the high computational cost of the existing solar radiation models.

In particular, there exist several GIS-based solar radiation models able to compute the quantity of solar energy that can be captured at a given terrain represented by a DEM. Solar Analyst [1] and SRAD [2] developed under ArcView GIS use simple methods to calculate direct and diffuse radiation maps but only on very small DEM and for small periods. The r.sun model [3] implemented under GRASS GIS environment [4] solves, using a numerical method, the empirical European Solar Radiation Atlas (ESRA) equations [5, 7] in order to calculate the clear-sky global irradiation maps considering the shadowing of the local terrain (i.e., produced by the DEM itself). Due to the computational limitations of the used shadow and radiation methods, this model can be applied only on small to medium DEM sizes. The high performance radiation model described in [16, 20] solves ESRA equations [5, 7] using a new scalable numerical method especially appropriate for current high performance computer architectures. The authors showed that their method is faster by many orders of magnitude than all the previously cited models for the same terrain sizes. This software is publicly available from [11]. This radiation model is fed with the shadows calculated by the multilevel horizon model developed by the same authors [19]. Both radiation and horizon models are able to address large terrains without any size constraints on high performance computing systems such as supercomputers and clusters of single or multicore sockets. The used data partition method as well as the scheduling policy are described in details in [19].

All the models described above consider the heights, slopes and aspects of the points of the land surface, i.e., the data provided by the DEM. However, they do not consider placing a surface at different heights, slopes and aspects over the points of the DEM. Recall that any modification in these parameters may changes considerably the received solar radiation. Few works in the literature have addressed applications affine to this problem but only on fix surfaces. In [8], the authors developed a tool called PVGIS based on r.sun [3] for estimating the photovoltaic potential on specific building surfaces, generally roofs with a predefined slope and azimuth. A similar tool called PV has been proposed in [9] to be implemented under ArcGIS. Both tools are limited to small urban areas due to the fact that the used sequential baseline radiation models *per se* are very expensive from a computational point of view and building new tools on the top of them makes them even more expensive. The characteristics of the software described above is summarized in Table 1.

Table 1: The difference between the proposed tool and related tools

GIS-Tool	Environment	DEM size	Target systems	Calclates	Horizons
Solar Analyst (public licence)	ArcGIS	small	single processor	solar radiation	local horizon
PV Analyst	ArcGIS	small	single processor	PV potential on predefined roofs of residential areas. Based on Solar Analyst.	local horizon
r.sun (public licence)	GRASS	small	single processor	solar radiation	local horizon
PVGIS	GRASS	small	single processor	PV potential on predefined roofs of residential areas. Based on r.sun	local horizon
Insolator (public licence)	Insolator	unlimited	multiprocessor	solar radiation	self+near+far horizons, without edge effect
Our tool	Insolator	unlimited	multiprocessor	maximum solar energy map, optimal slope map and optimal orientation map. Based on Insolator	self+near+far horizons, without edge effect

In this work, we have developed a new tool able to compute the maximum solar energy that can be collected at each solar module situated on each point of a vast high resolution DEM and the optimal slope and orientation maps that allow capturing that maximum solar irradiation during different periods of time. In particular, we have analyzed three high performance methods on GPU and CPU based systems: (1) an exhaustive method used for comparison

purposes, (2) a divide and conquer algorithm and, (3) a gradient ascent algorithm. The calculated maximum solar energy and optimal angles maps show coherent results. The computational results show that the irregular load of the gradient ascent method makes it more suitable for multicore systems while the regular load of divide and conquer algorithm is more appropriate for GPUs systems. The findings presented in this paper can be easily incorporated into GRASS GIS [4] and ArcView GIS [4] environments.

This paper is organized as follows. The baseline algorithms, i.e., the irradiation and horizon algorithms, and their adaptation to calculate the maximum solar irradiation are described in section 2 and 3 respectively. A description and analysis of the three studied methods is provided in section 4. The parallel implementation and optimizations are given in section 5. The numerical and performance results are described in section 6 and 7 respectively. Finally conclusions are summarized in section 8.

2. Horizon Computation

The horizons also called shadows must be generated before the irradiation computation. The multi-level composition algorithm we developed in a previous work [19] has been used as kernel. This algorithm was designed to take advantage of multiprocessor computers, which in consequence allowed us to increase the accuracy of the results and overcome the memory limitations. This method calculates the total horizon of a given point as the maximum elevation of three components: (1) an exact ground horizon, (2) a high-precision near-end horizon and, (3) a low-precision far-end horizon. These three horizons are computed separately using the appropriate DEM resolution as follows. Firstly, the space around each point of the terrain is divided into $s = 360$ sectors, where $\hat{s} = 1^\circ$, and then the horizons of all the points are calculated sector by sector till sweeping the entire 360 sectors. Upon completion, each point of the terrain will get an array of 360 elevation angle values. The optimizations of this algorithm for single and multi-processors are described in details in [19].

A brief description of each one of these components together with an analysis of their adaptation to calculate the maximum energy on a solar module can be summarized as follows.

- A horizontal surface represented by a point in the DEM views at most one half of the sky dome because the ground hides the other half. On inclined surfaces, there is an additional fraction of the sky dome that is not visible from the points that belong to that surface due to self-hiding, which we call ground horizon. This component is calculated in each sector using this formula, $hor_g = \max(0, \alpha - 90^\circ)$, where α is the angle between the normal to the surface (i.e., the point of the DEM) and the search direction \vec{u}_s .

Usually the center of solar panel is placed at an altitude on the order of 1 m above the ground surface. In most cases, this height is negligible compared with the resolution of the DEM. In consequence the panel self shading must be added to the ground horizon. However, if the resolution of the used DEM is comparable with the panel altitude then the ground horizon must be replaced by the panel self horizon.

- The near horizon produced by the closest points in the terrain is calculated using the full resolution DEM that fits entirely in the RAM memory. If the DEM does not fit in the memory hierarchy, a block partition strategy can be applied [19]. Placing a solar panel at a given altitude affects slightly this horizon if and only if the precision of the DEM is comparable with the size of the panel. Generally, this component is not required to be recalculated.
- The far-horizon is calculated using a low resolution halo DEM, that surrounds the high resolution one taking into account the elevations placed behind a certain limit, roughly 1 Km to 10 Km. This limit depends on the desired precision and the memory hierarchy capacity. The fact of placing the solar panel at a certain altitude does not affect at all this horizon.

3. Solar Irradiation Computation

The algorithm used to compute the solar irradiation that can be captured at each pixel of the DEM can be summarized as follows:

- The sky is discretized and the sun trajectory is calculated. The number of times the sun has passed from each pixel of the sky dome is calculated for all the considered time period. This calculation is done only once for all the points of the terrain.
- Using the previously calculated data, the energy matrices, N_1 , N_2 , N_3 , N_4 , N_5 and, N_6 are calculated on the considered area for a discrete set of horizon elevations, azimuths and, heights. These matrices, which are described in [16, 20], are shared between all the points of areas of similar atmospheric and astronomical conditions, usually areas of up to 100 Km². The calculation at this stage does not depend on the DEM, it only needs the global latitude, longitude and turbidity of the considered region.
- The global solar irradiance, $G_{ic}(Wm^{-2})$, that can be captured at point P of the DEM with slope, also called inclination, γ_N , azimuth, also called aspect A_N and, height, also called elevation z , is calculated using the precomputed horizon as follows.

```

For each point P of the terrain
/*  $A_N$  and  $\gamma_N$  must be known at this stage */
For  $A_0 = 0, 360^\circ - 1^\circ$ 
/* The near_hor() and far_hor() are already calculated */
hor=max(ground_hor(),near_hor(),far_hor())
val1+= $N_1(z,A_0,hor)$ 
val2+= $N_2(z,A_0,hor) \cdot \cos(A_0-A_N)$ 
val3+= $N_3(A_0,hor)$ 
val4+= $N_4(z,\gamma_N,A_0,hor)$ 
val5+= $N_5(z,A_0,hor) \cdot \cos(A_0-A_N)$ 
val6+=hor/90
End For
 $G_{hc}=val1+val3 \cdot F(\gamma_N)+val4$ 
 $G_{ic} = k_c \cdot (\cos(\gamma_N) \cdot val1 + \sin(\gamma_N) \cdot val2 + val3 \cdot F(\gamma_N) + val4 + \sin(\gamma_N) \cdot val5 + \rho_g(x,y) \cdot G_{hc} \cdot val6/360)$ 
End For

```

Where val_i with ($i = 1, 6$) are auxiliary variables; h_0 and A_0 are the altitude and azimuth of the sun respectively. $F(\gamma_N)$ is a trigonometrical expression to consider the non-circumsolar component of the diffuse radiation. k_c is the clear-sky index, which represents the attenuation of the irradiation due to clouds. The global irradiation $G_{ic}(Wm^{-2})$ can be calculated as the sum of the irradiance during a specific time interval on the order of minutes, hours, days or years.

4. Maximum Solar Energy and Optimal Slope-orientation Maps Computation

We have developed three methods to compute the maximum solar energy and optimal angles maps: (i) the global maximum algorithm, used only for comparison purposes since it is very costly from a computational point of view, specially for vast DEMs, (ii) the divide and conquer algorithm and, (iii) the gradient ascent algorithm. The three proposed methods are based on the horizon and irradiation kernels described in section 2 and 3.

4.1. Global Maximum Algorithm

The first algorithm is a naive method consists of exhaustively finding, for each point of the terrain, the combination (slope γ_N , orientation A_N) that captures the maximum solar irradiation. This calculation can be carried out by sweeping all possible 90 slope and 360 orientation angles, 1° by 1° . The found optimal combination corresponds to the maximum irradiation input that can be captured at that surface. The computational cost of this method can be reduced by considering only the slope and orientation intervals that receive direct solar irradiation from the east to the west passing by the south. In fact, we experimentally found that the optimal slope and orientation in most regions of the northern hemisphere belong to $[30^\circ, 75^\circ]$ and $[90^\circ, 270^\circ]$ respectively. Notice that this algorithm is regular since it iterates over the same number of solutions for each point.

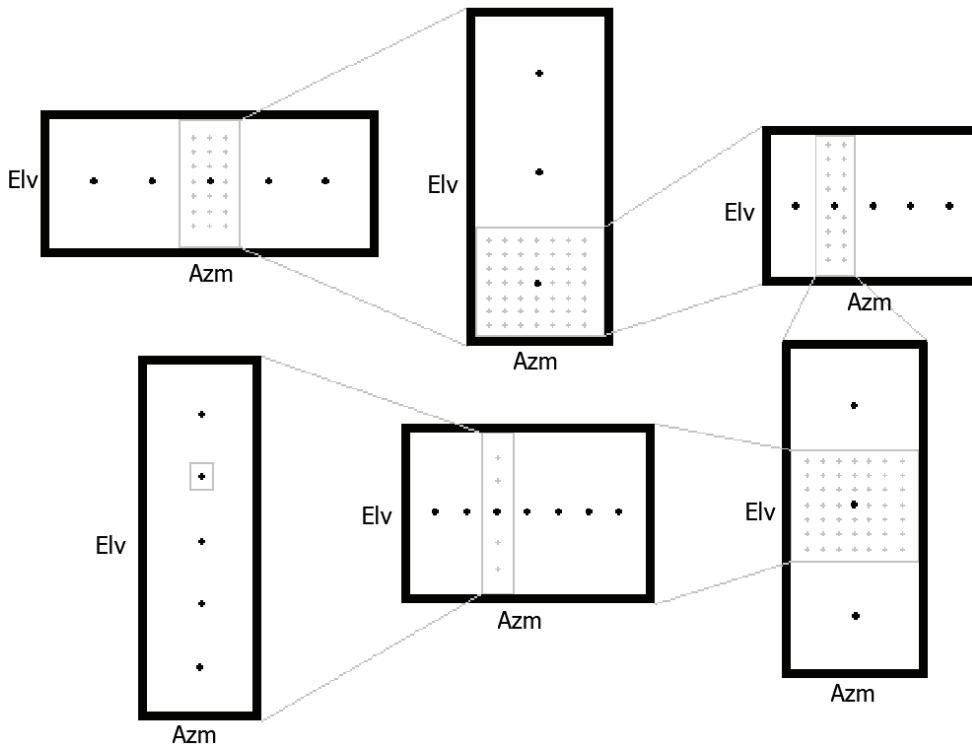


Figure 1: Data partition of the orientation (Azim) and slope (Elv) angles in divide and conquer method.

4.2. Divide and Conquer Algorithm

For each point of the DEM, this method divides the space (γ_N, A_N) into n regions and evaluates the calculated irradiation at their centers. Afterwards, it iterates over the neighborhood of the center with the maximum energy till it reaches again the maximum value. Successive iterations will lead to the slope and orientation in the solution space, which will be considered as the optimal solution.

This method applies successive partitions in slope and orientation. Experimentally, we found that odd numbers of partitions ensures more accurate solutions for the majority of the points. We also found that carefully selecting the prime factorization of the slope and orientation intervals reduces the total number of iterations. That is, the slope interval, $[30^\circ, 75^\circ]$, has $75 - 30 + 1$ slope values; the nearest value to this number (46) whose prime factorization gives the smallest addition of factors is 45 ($3 \times 3 \times 5$). Consequently the range should be changed to $[31^\circ, 75^\circ]$. Specifically, we performed a partition of $5 \times 5 \times 7$ in orientation and $3 \times 3 \times 5$ in slope as can be seen in Figure 1. Where, first the orientation interval is partitioned into 5 blocks, then the slope interval of the block with the largest value of irradiation is partitioned into 3 blocks and successively the block with the maximum value of irradiation in the current partition is partitioned into 5 blocks and so forth.

This algorithm is substantially faster than the global maximum algorithm. Nevertheless, for some points, about 6%, the found maximum value is not a global maximum. A further analysis showed that the difference between the local and global maximum irradiation received by this set of points is not considerable; in addition, these points receive few irradiation input, which means that they are not candidate for placing solar panels.

The computational load required for each point of the DEM is similar, which make this algorithm also regular.

4.3. Gradient Ascent Algorithm

The third algorithm consists of starting from an initial value of slope and orientation, e.g., $\gamma_N = 34^\circ$ and $A_N = 180^\circ$, and moving in azimuth in the direction of the fastest grow irradiation values, until a maximum is reached. Then, using that value of the azimuth, the algorithm moves in orientation in the direction of fastest grow of irradiation values.

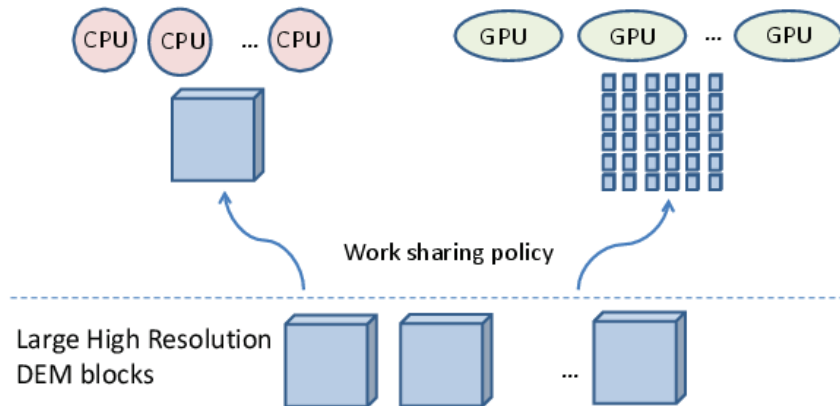


Figure 2: The work-sharing scheduling policy used in our parallel implementation for GPU-CPU systems.

Similarly to the second algorithm, this method stops the search when a maximum is reached. The obtained values of energy, slope and orientation will be considered as the optimal values. For few points the global maximum may not be reached and consequently the found value would be a local maximum. Usually, these points are not candidate for installing solar panels.

The number of iterations carried out by this method varies from point to point of the terrain. The experiments carried out in this work shows that each point of the DEM needs an average of 7.74 iterations per point.

5. Parallel implementation and optimizations

The calculation of the irradiation at a given point of the terrain does not require any information from the points of the neighborhood. Therefore, we have implemented our algorithm using work-sharing data-parallel model based on block partition (see Figure 2). Where first, the terrain is partitioned into blocks that fit entirely in both, the shared memory of multicore system and the global memory in of the GPU. Each block can be assigned to either an available CPU or GPU. The cores of the multicore system work on a pool of blocks, while the cores of the GPU work on a pool of points of the same block. That is, each thread in the multicore system picks up one block at time from the pool of blocks and processes it individually, while threads running on the same GPU pick up one block at time from the pool of blocks and work all together to process all the points of that block. The parallel implementation can be sketched as follows:

```

/*Each iteration is mapped only on one CPU or one GPU */
For each block of the terrain
/*These iterations are mapped only on GPU cores */
  For each point P of the block
    /*Find  $G_{icmax}$  and optimal  $A_N$  and  $\gamma_N$  using
    divide and conquer or ascent gradient methods */
    computeANDget_max( $G_{icmax}, A_N, \gamma_N$ )
  End For
End For

```

Several optimizations have been applied to make the parallel implementation more efficient on both CPU and GPU.

5.1. Arithmetic Operations Optimizations

The calculation of the maximum solar irradiation at each specific combination (γ_N, A_N) implies repeatedly re-computing $\sin(\gamma_N)$, $\cos(\gamma_N)$ and, $\cos(A_0 - A_N)$, which are expensive in terms of CPU utilization. This calculation can be optimized by storing in arrays all the possible values of $\sin(\gamma_N)$, $\cos(\gamma_N)$ and, $\cos(A_0 - A_N)$ and reading them whenever it is necessary.

5.2. Data Accesses Optimizations on GPU

The GPU has 6 types of memories, registers, local memory, shared, global, constant and, texture memories. Some of them are accessible by the programmer and some can be accessed only by the compiler. An appropriate mapping of data in these memories is essential for the performance. Our algorithm deals with these data structures: horizon, energy matrices, pre-calculated arrays of $\sin(\gamma_N)$, $\cos(\gamma_N)$, $\cos(A_0 - A_N)$ arrays and, the resulting maps.

The horizon is an array that stores the elevations seen in 360° of each point of the terrain. For a block of 500 points, we need to allocate 180000 bytes for the horizon. This information is too large to fit in registers or shared memory. It can only fit in global memory. Thus, first the horizon is allocated in the global memory, then each thread will make a copy of the horizon of the point it will process into the local memory. These accesses should be synchronized to ensure that all the threads access the same local position at the same time.

The energy matrices are read by all the threads to compute the irradiation for all the points. Due to their high storage requirement, about 24 Mb, and irregular accesses, it is appropriate to allocate them in the global memory.

The calculated trigonometric values stored in arrays can fit in the shared memory. However, in this memory, as threads are synchronized, all the simultaneous accesses to the same values are serialized. Thus, we keep $\sin(\gamma_N)$, $\cos(\gamma_N)$ arrays in the shared memory and allocate the $\cos(A_0 - A_N)$ array in the local memory to get advantage of the aligned accesses.

We have not mapped data into constant and texture memories due to their low capacity and synchronization constraints.

6. Performance Results

Table 2: Experimental testbed

	CPU AMD Athlon 64 3000+	GPU NVIDIA GeForce GTX 460
# Cores	1	336
Clock Speed	1.8 Ghz	1.43 Ghz
Main Memory	2 Gb DDR 266 Mhz	1 Gb GDDR5 1800 Mhz
Memory hierarchy	L1: 64K + 64K L2: 512 K	32762 Registers 48 Kb Shared L2: 524 K

The parallel implementation of our algorithm has been carried out using C++ programming language. To create and manage threads on the GPU and CPUs we have used NVIDIA CUDA [14, 13] and openMP [15] respectively. This parallel code is available via email from the corresponding author.

A comparison of the runtime and speedup of the parallel implementation when using each one of the three methods, on a DEM of 500 points and the CPU and GPU, is shown in Table 3. For each method, we performed 10 executions and took the average runtime.

Table 3: Runtimes and speedups of each one the three methods on CPU and GPU using a DEM of 500 points.

	Runtime on CPU (in ms)	Runtime on GPU (in ms)	speedup
Global Maximum	163980,21	6957,33	23,57
Divide and Conquer	600,63	21,31	28,18
Gradient Ascent	128,01	24,77	5,17

As it can be seen in Table 3, the gradient ascent algorithm is the most efficient method on CPUs since it gives a substantially smaller runtime than the exhaustive and divide and conquer methods. However, the divide and conquer algorithm overcomes both the exhaustive and gradient ascent algorithms on GPUs showing a good scalability. This can be explained by the fact that divide and conquer has a well balanced load among all the points of the terrain which

makes it specially appropriate for GPUs. Whereas, the gradient ascent algorithm has a low but irregular load which makes it more suitable for multicore systems.

The scalability of the parallel implementation that uses the gradient ascent on CPUs and divide conquer on GPUs is a linear function of the number of cores and GPUs since no communication is required neither between threads that work on different blocks nor between threads that work on the same block.

7. Numerical Results

The numerical correctness and accuracy of the proposed high-performance algorithm is showed in two DEMs of different resolutions. As it can be verified in Figure 3, the north-western faces of some mountains of the region of Malaga captures substantially less solar irradiation due to the shadow in the south produced by these mountains themselves. Constant values of irradiation (7800 MWhm^{-2}), slope (34°) and orientation (180°) correspond to hypothetical panels at the sea level. Similarly in Figure 4, the shadow of the buildings affects substantially the quantity of the energy that can be produced in nearer locations.

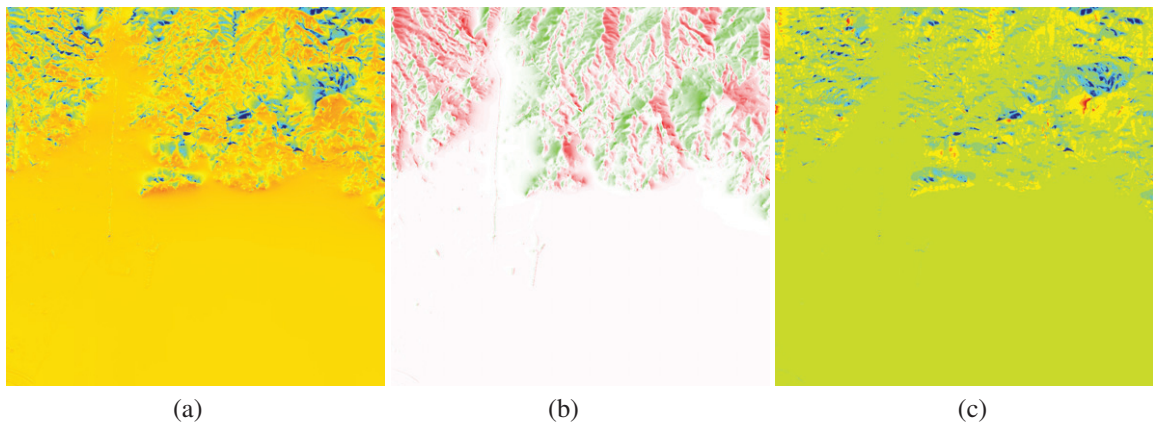


Figure 3: (a) The annual maximum solar energy map that could be captured in the region of Malaga, where the blue and red colors correspond to 6000 MWhm^{-2} and 7800 MWhm^{-2} respectively. (b) optimal orientation map of the same region, where blue and red colors correspond to the west and east respectively. (c) optimal slope map of the same region, where blue and red colors correspond to 30° and 50° respectively.

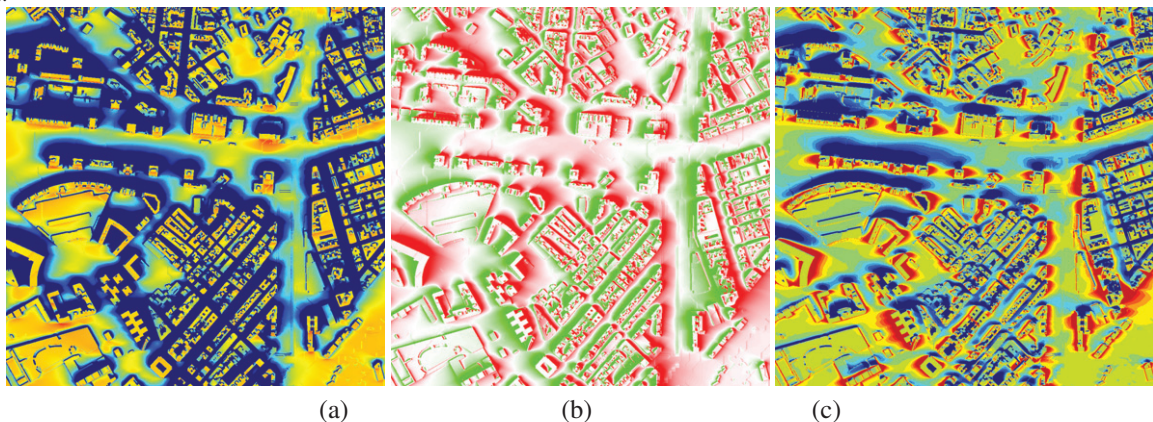


Figure 4: (a) The annual maximum solar energy map that could be captured at an Urban area of Malaga, where the blue and red colors correspond to 6000 MWhm^{-2} and 7800 MWhm^{-2} respectively. (b) optimal orientation map of the same area, where blue and red colors correspond to the west and east respectively. (c) optimal slope map of the same area, where blue and red colors correspond to 30° and 50° respectively.

8. Summary and Conclusions

This paper has presented a new fast, accurate, and parallel algorithm able to compute the maps of the maximum solar irradiation input and the optimal angles, appropriate for large high-resolution DEMs and heterogeneous computer systems such as GPU-CPU platforms. The proposed algorithm is useful for many professionals in academia and industry. The implementation described in this article represents a powerful tool necessary to compute the map of the maximum solar irradiation with a high accuracy, especially for areas with stable climatic conditions. Actually, it can be easily included in Geographical Information Systems (GIS) tools such as ArcGIS [6] and GRASS [18, 12] environments.

References

- [1] P. Fu and P.M. Rich, The solar analyst 1.0 User Manual, 2000.
- [2] J. P. Wilson and J. C. Gallant (Editors), Secondary topographic attributes, Terrain Analysis: Principles and Applications (John Wiley & Sons, New York, 2000).
- [3] M. Suri and J. Hofierka, Trans. in GIS 8 (2004) 175.
- [4] GRASS Development Team, Geographic Resources Analysis Support System (GRASS) Software, 2007, <http://grass.itc.it>.
- [5] K. Scharmer and J. Greif (Editors), The European Solar Radiation Atlas: Volume 2, Database and exploitation Software, (Les Presses de L'ecole des Mines, Paris, 2000).
- [6] ESRI 2011. ArcGIS Desktop: Release 10. Redlands, CA: Environmental Systems Research Institute.
- [7] C. Rigollier, O. Bauer and L. Wald, Solar Energy 68 (2000) 33.
- [8] J. Hofierka, J. Kanuk, Assessment of photovoltaic potential in urban areas using open-source solar radiation tools, Renewable Energy (2009). Volume 34, Issue 10, Pages: 2206-2214.
- [9] Y. Choi, J. Rayl, C. Tammineedi, J. R. S. Brownson, PV Analyst: Coupling ArcGIS with TRNSYS to assess distributed photovoltaic potential in urban areas, (September 2011) doi:10.1016/j.solener.2011.08.034 Key: citeulike:9772871.
- [10] S. Bari, Optimum slope angle and orientation of solar collectors for different periods of possible utilization, Energy Conversion and Management (2000) Volume 41, Issue 8, Pages 855-860.
- [11] www.ac.uma.es/~siham/research/radiation.tar.gz
- [12] GRASS Development Team, Geographic Resources Analysis Support. System (GRASS) Software, <http://grass.itc.it>, 2007.
- [13] NVIDIA CUDA C best practices guide.
- [14] NVIDIA CUDA C programming guide.
- [15] <http://openmp.org/wp/openmp-specifications/>
- [16] L.F. Romero, S. Tabik, J. Vias, Emilio L. Zapata. Fast clear-sky solar irradiation computation for very large digital elevation models. Computer Physics Communications , 178(11), 800-808, 2008.
- [17] SODA: Services for Professionals in Solar Energy and Radiation, <http://www.soda-is.com/eng/index.html>.
- [18] M. Suri, J. Hofierka, Trans. in GIS 8 (2004) 175.
- [19] S. Tabik, L.F. Romero, E. L. Zapata. High Performance Three-horizon Composition Algorithm for large scale terrains, International Journal of Geographical Information Science , 25(4), 541-555, 2011.
- [20] S. Tabik, J. M. Vias, E. Zapata, L.F. Romero: Fast Insolation Computation in Large Territories. International Conference on Computational Science (1), 54-61, 2007.